

Software Engineering Basics

Part 1: Introduction & Distributed Version Control

Part 2: Automated Testing and Continuous Integration

Institute for Software Technology

Margrit Klitz, Daniel Abele

Moritz Hof, Martin Kühn, Jonas Thies, ...

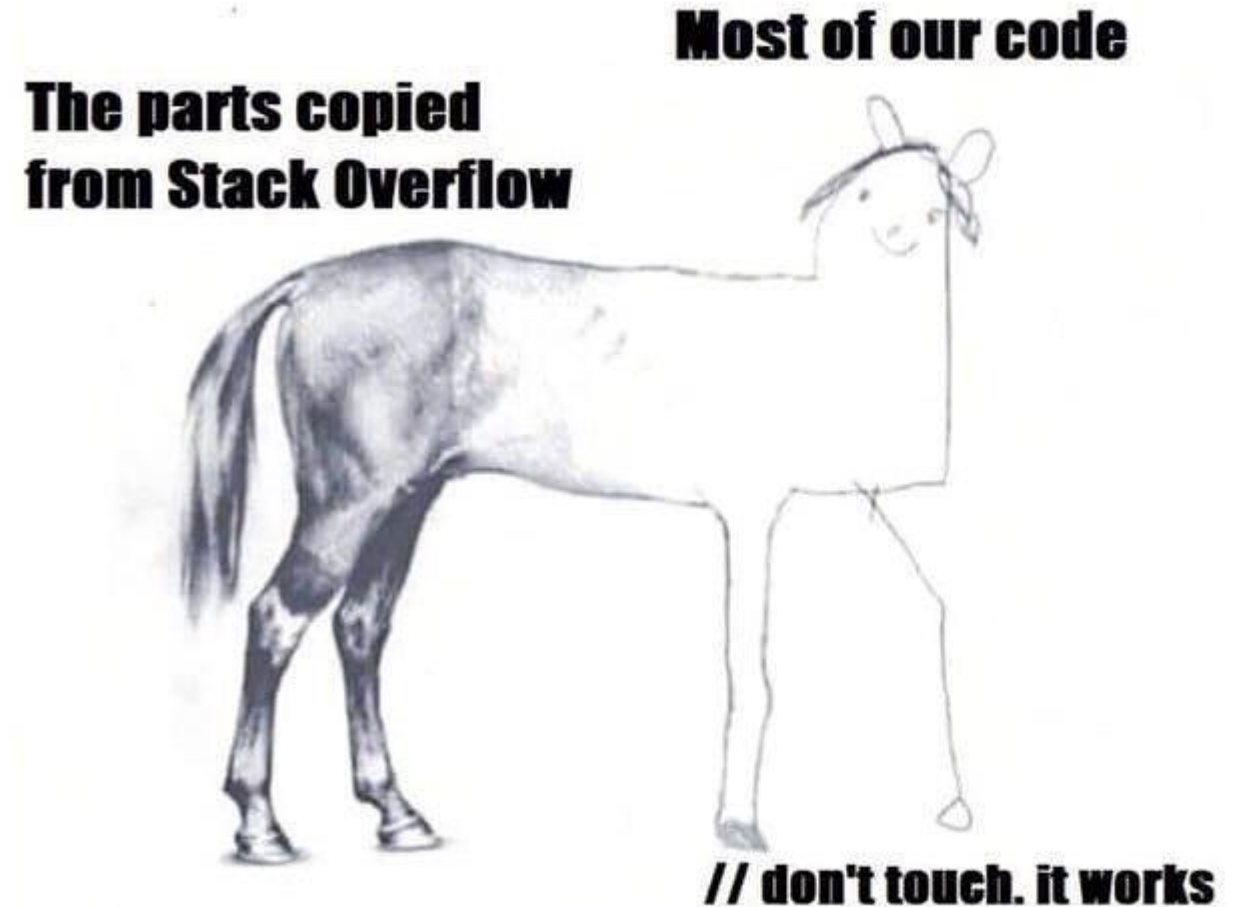


Knowledge for Tomorrow



Today

- SC & SC-HPC
- Software Engineering
 - What's out there
 - Version control (DLR-Gitlab)
 - Tests
 - Continuous Integration
- Aim
 - Get to know the HPDA support team
 - Learn something helpful
 - For us: Where to dig deeper



Margrit Klitz

- Mathematics at Bonn University
- **In DLR** since August 2015
 - Parallelization
 - Mesh and data management for flow simulations
 - Software engineering topics
 - Deputy head of the High-Performance-Computing department



Institute for Software Technology

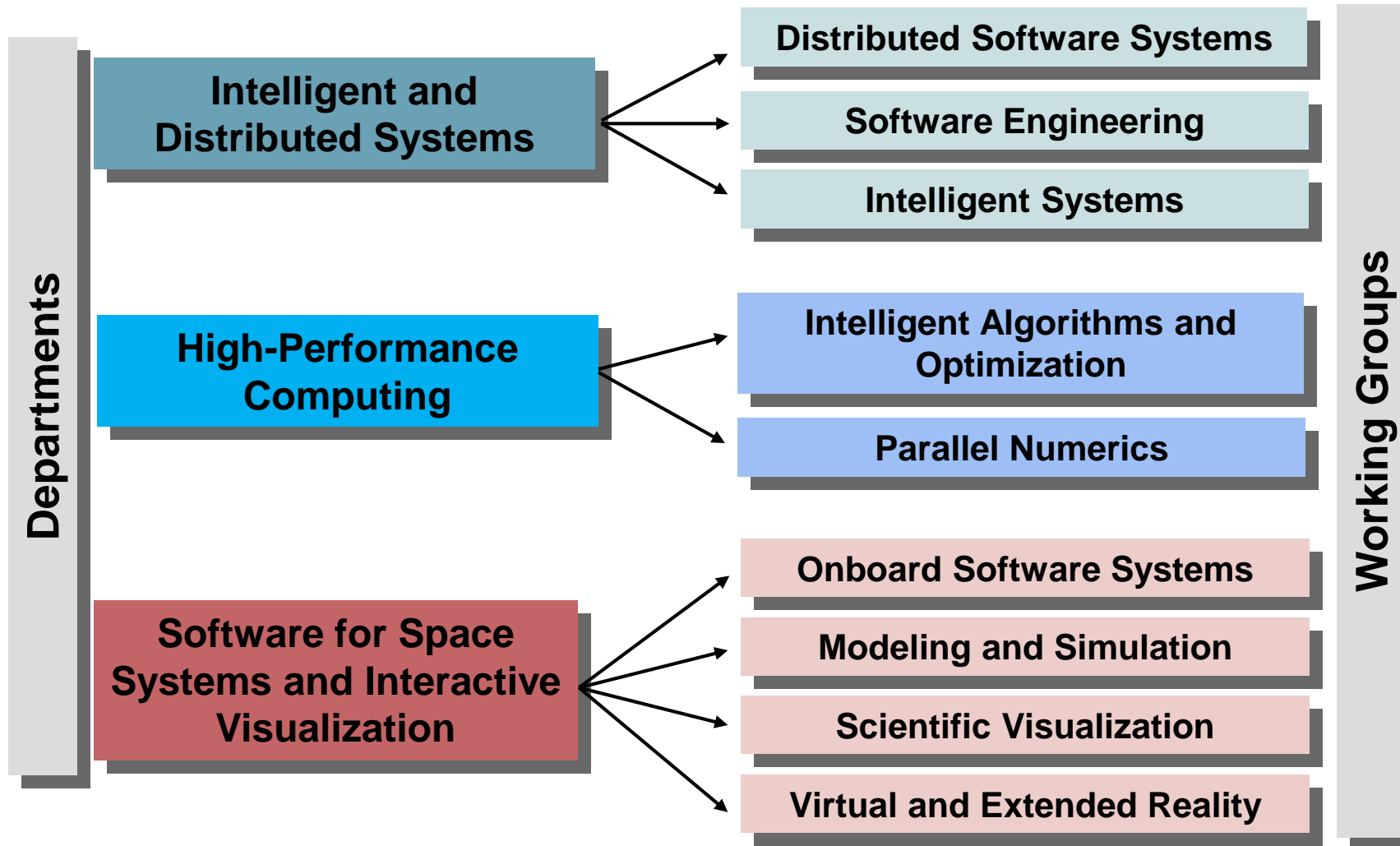


- Steht für **innovatives Software Engineering**,
- Entwickelt **anspruchsvolle Individualsoftwarelösungen** für das DLR und
- Ist Partner in **wissenschaftlichen Projekten** im Bereich Simulations- und Softwaretechnologie



DLR Institute Simulation and Software Technology

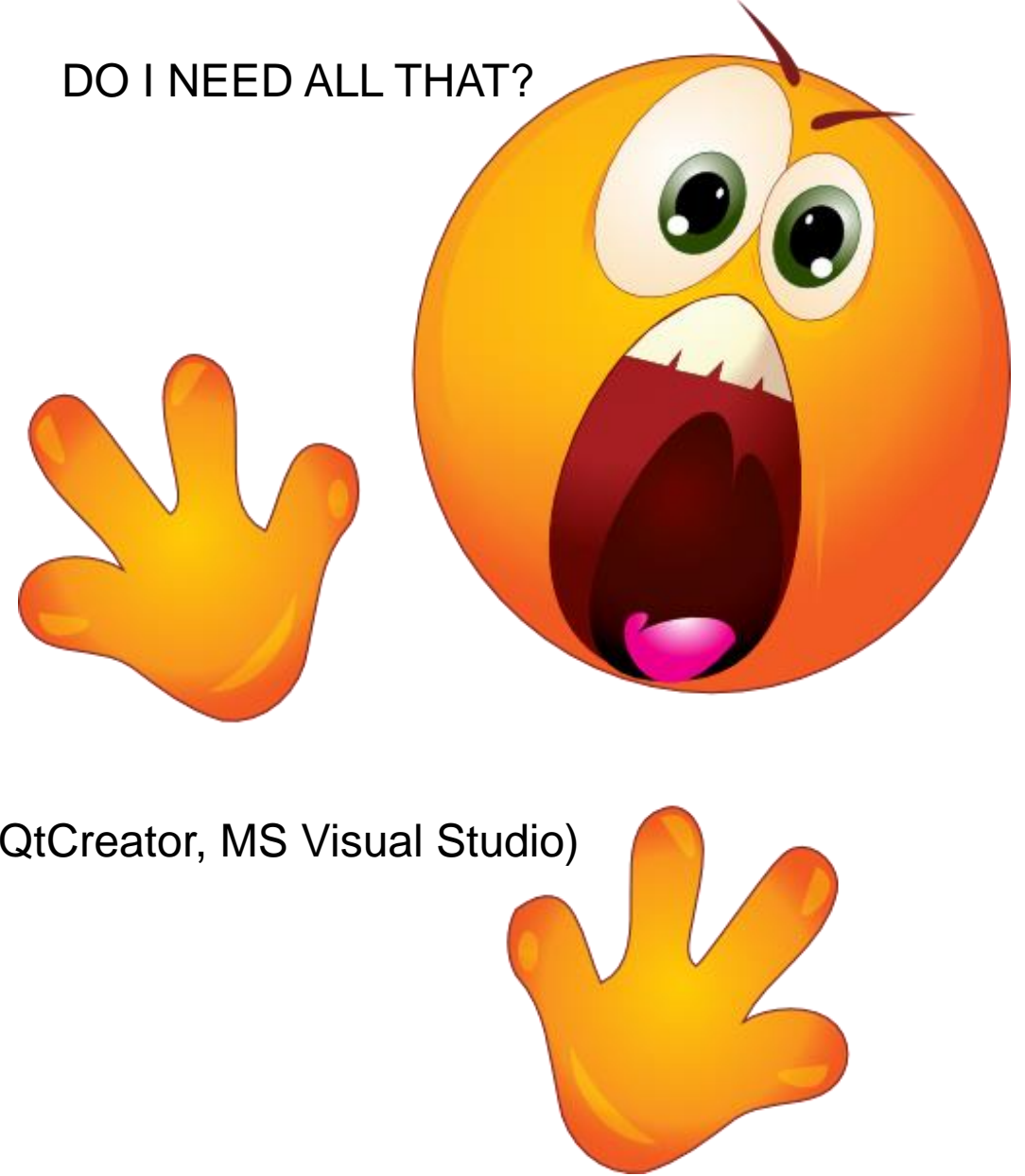
Scientific Themes and Working Groups



Aspects of modern software development

- Distributed development processes via git, subversion,...
- Community software (github, bitbucket,...)
- Open source licensing (BSD, MIT, (L)GPL,...)
- Software architecture (z.B. UML for graphical description)
- Build systems (CMake, Autotools,...)
- Meta build systems (Spack, EasyBuild, Conda)
- Test frameworks (GoogleTest, PyTest, junit, ...)
- Continuous integration testing (Jenkins, gitlab-ci,...)
- Integrated development environments (IDEs, e.g. Eclipse, QtCreator, MS Visual Studio)

DO I NEED ALL THAT?



Probably not. But some of it may be **very useful**

At DLR we categorize software in order to come up with a reasonable subset for each Individual software effort:

Class 0: short scripts, mostly private use, purpose: try something out, generate plots for a paper etc.

Class 1: prototypical software that should be used and extended by others

Class 2: Software intended for long-term use also outside the own group

Class 3: critical software or software with product character



In this talk – The most basic but very useful parts

- Version control: DLR-Gitlab
- Automatic Testing: e.g. GoogleTest
- Continuous Integration: e.g. Jenkins



Version control – why and how SIMPLY EXPLAINED



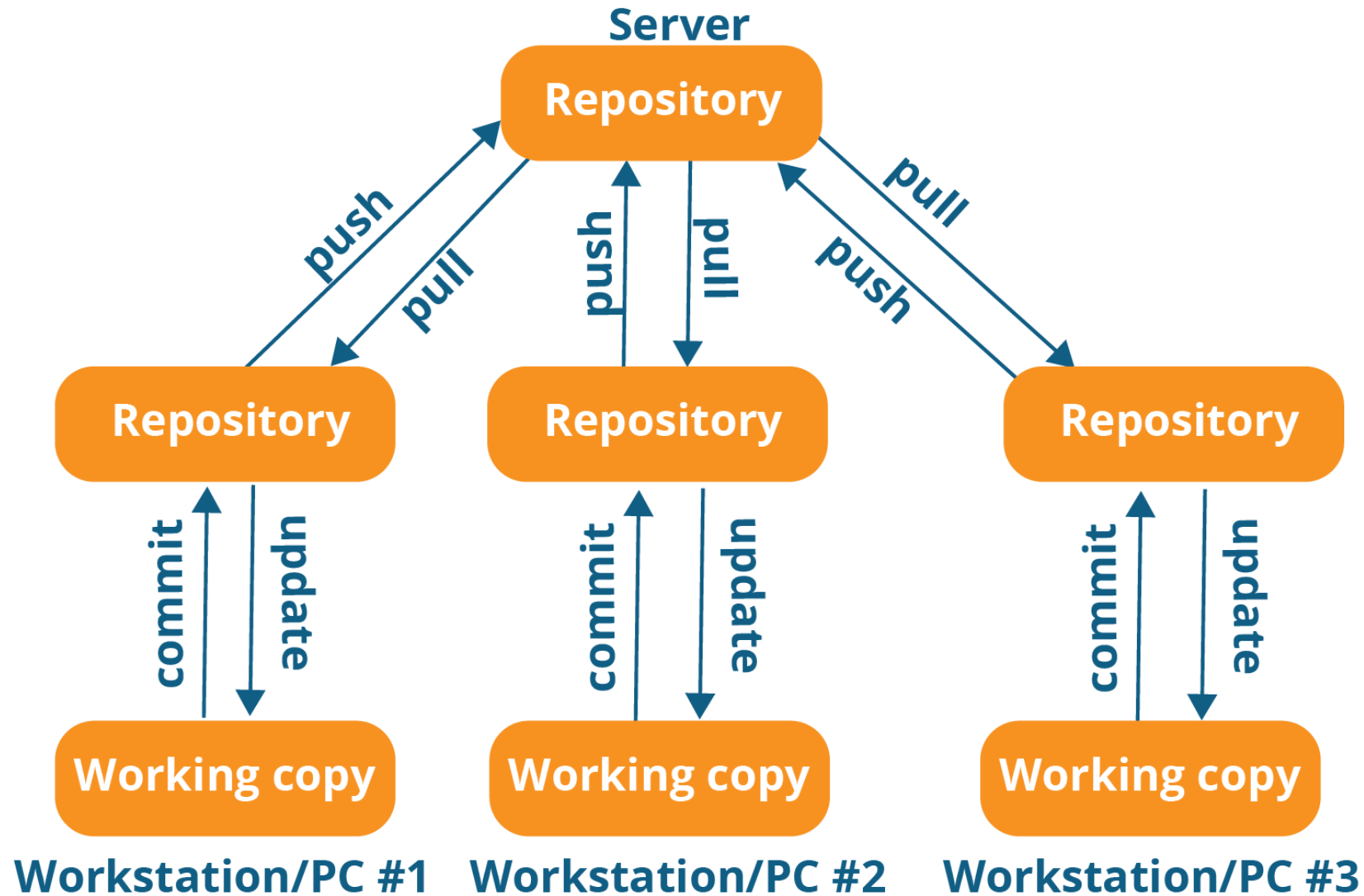
- Simple case: version = sequence changes:



<https://aberdienstudygroup.github.io/studyGroup/lessons/SG-T1-GitHubVersionControl/VersionControl/>

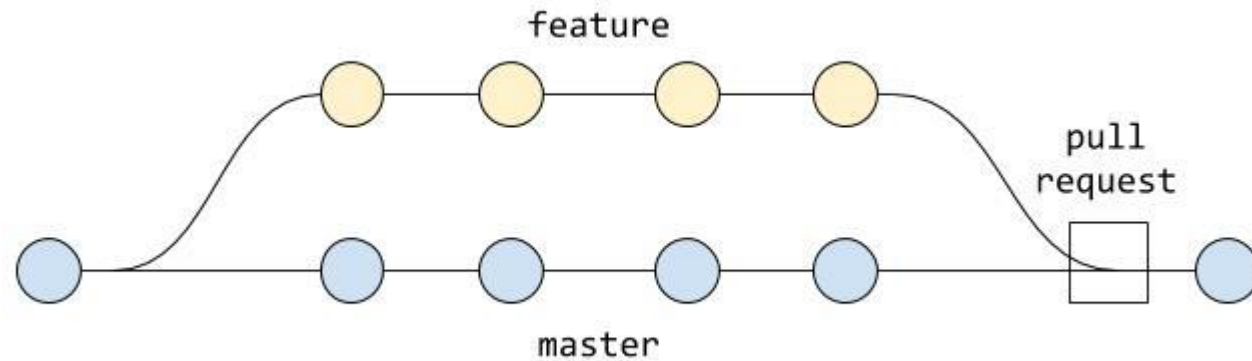


Distributed version control system



Practical Introduction

- Setting up a project
- Cloning
- Making changes
- Push/Pull
- Feature branch
- Issues/Kanban



Vielen Dank für die Aufmerksamkeit!

Fragen?

Dr. Margrit Klitz

German Aerospace Center (DLR)

Simulation and Software Technology

Department High Performance Computing

Margrit.Klitz@dlr.de

<http://www.DLR.de/sc>

